EMBEDDED OPEN SOURCE SUMMIT

EMBEDDED LINUX CONFERENCE

Debugging Heterogenous SoC using OpenOCD

2024-04-18, Seattle

Nishanth Menon
Jason Peck

Texas Instruments

About us: TI Processors and Open source

Decades of contribution and collaboration

Ingrained culture to give back to the community

Upstream FIRST!

Focus on long term, sustainable and quality products

Upstream and opensource ecosystem in device architecture

Upstream FIRST mentality!

Authors

Nishanth Menon, *Senior Member Technical Staff, Texas Instruments*

Nishanth has been working with TI for over 18 years. He is part of TI's Embedded Processing organization as a Linux/Systems architect working with the Jacinto and Sitara Processor families. He is also a long-time Linux kernel contributor and currently an active Linux kernel maintainer for TI's K3 device trees, among other things he does.

Jason Peck, *SoC Architect, Senior Member Technical Staff, Texas Instruments*

Jason has been with TI for over 25 years, spending much of that time in various roles related to the architecture and design of the full spectrum of on-chip debug capability deployed within TI cores, accelerators, and devices. He actively participates in standards bodies, including IEEE and MIPI Alliance.

Disclaimers

- This is a technology presentation, not product-readiness or roadmap commitment
- Opinions presented here are that of the speakers and may not reflect that of Texas Instruments Inc.
- I am not, by any stretch of the imagination, an expert in OpenOCD or JTAG (aka, please be gentle).
- The scope is limited to Arm debugging; DSP (C7x/C6x) debugging is theoretically possible but let us keep the conversation to Arm processors.
- This presentation assumes a level of embedded debug expertise, I give a few hints to help.

Overview

- Motivation
- Introducing basics
  - SoCs with Heterogenous Processors
  - JTAG
  - Arm CoreSight
  - OpenOCD
- Let us talk self-hosted debug with OpenOCD and dmem
- Quick peek into OpenOCD dmem internals

Motivation

- I am lazy and don't like to get off my couch to get to a JTAG adapter :)

AND

- I wanted the uC to blinken a LED :((apparently, it is still a multi-year project: Ref LPC 2023 talk)

AND

- I wanted to debug uC like Linux developers do – `gcc hello.c; gdb a.out`

AND

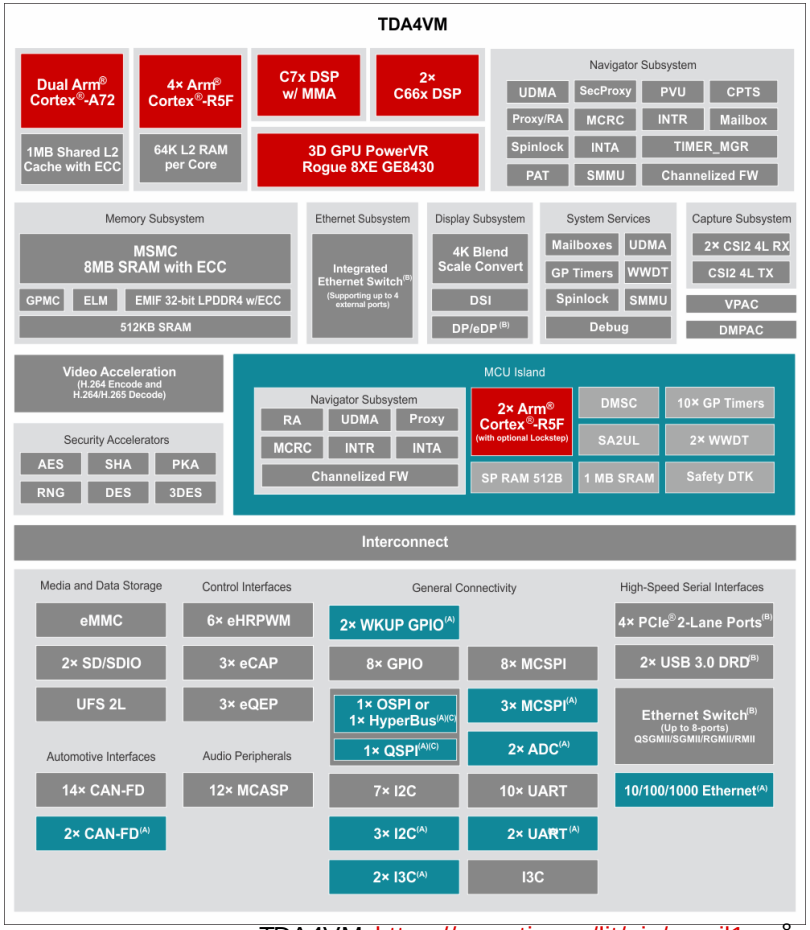
- It so happened that we wanted to do a project for a 24-hour hackathon competition at TI in 2022. (PS: The project won the hackathon)

Please don't make me a meme, acting sleepy after long winter night of patch reviews, CC-SA 4.0, Nishanth Menon

Introducing Basics

SoCs with Heterogenous Processors

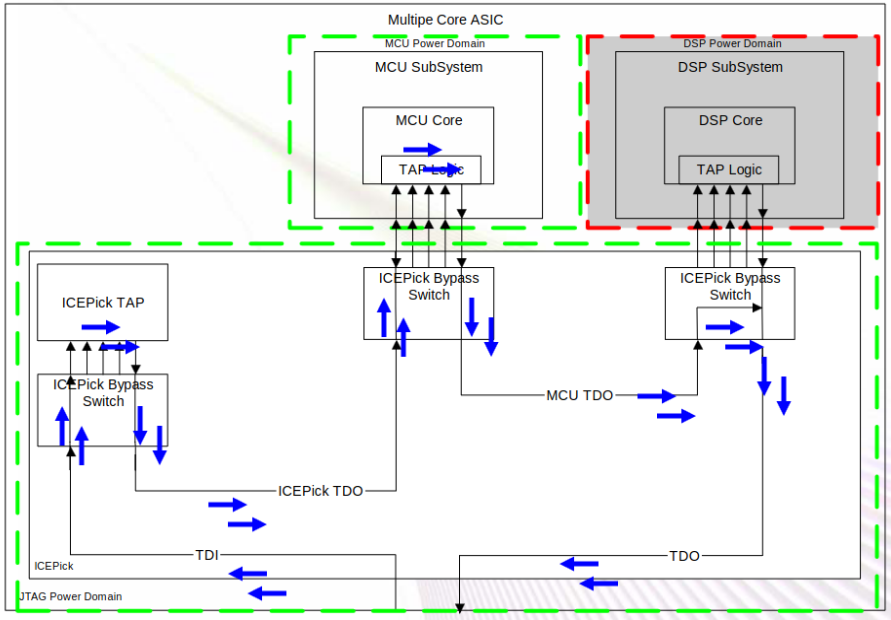
- Wikipedia definition: **Heterogeneous computing** refers to systems that use more than one kind of processor or core. These systems gain performance or energy efficiency not just by adding the same type of processors but by adding dissimilar coprocessors, usually incorporating specialized processing capabilities to handle particular tasks.
  - **Cortex-A:** Application processors typically run HLOS.
  - **Cortex-R:** Real-time hardware execution running RTOS.
  - **Cortex-M:** Power-efficient uC alternative running RTOS.
  - A sprinkle of specialized hardware accelerators.
- All these work simultaneously and communicate among themselves.
- **This complex problem is precisely what the OpenAMP group attempts to standardize, generalize, and simplify. Join us!**

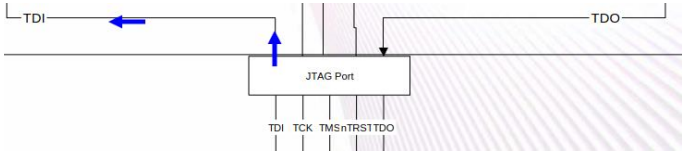


TDA4VM https://www.ti.com/lit/zip/spruil1

JTAG Intro

- Joint Test Action Group: Excellent intro by Aliaksandr Part 1 and Part 2

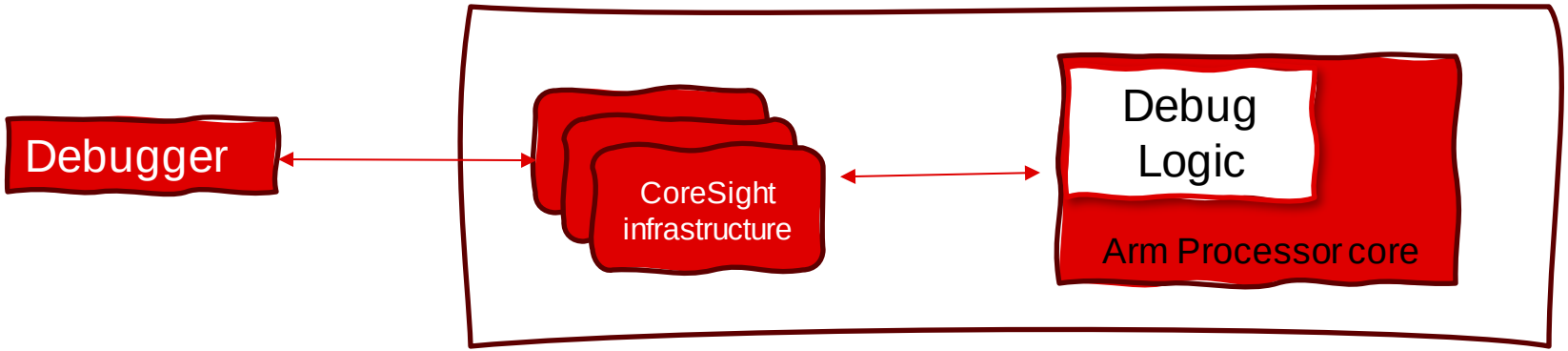




https://www.ti.com/lit/ml/sprp603/sprp603.pdf : ICEPick is TI's Test Access Port router (TAP router)

ARM CoreSight

- CoreSight is the on-chip debug and trace technology that Arm provides for multi-core processor SoC architecture.
- Refer to ARM Documentation, Coresight architecture, ADI V5 specification and training documentation
- Other informative links: Linux Kernel documentation and youtube presentations

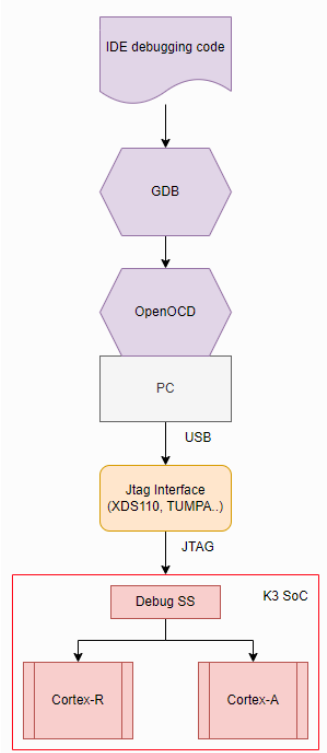


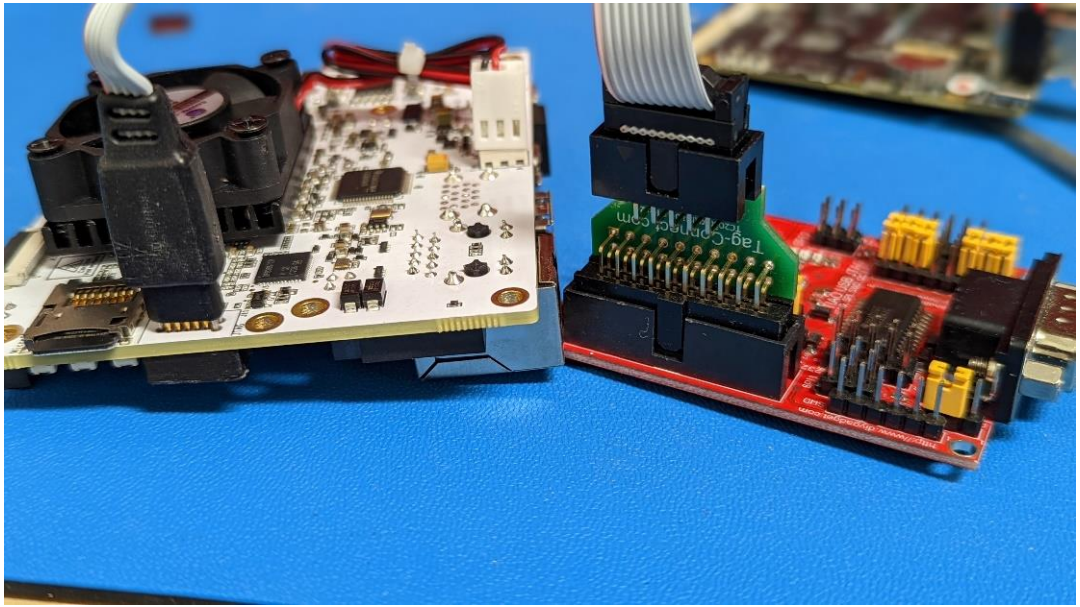
Disclaimer: above diagram is a redraw of from Arm Ltd. site since I wasn't sure of the copyrights

OpenOCD Intro

- OpenOCD (Open On-Chip Debugger) is an open-source project that supports debugging and in-circuit programming for various microcontrollers and microprocessors. It allows developers to interact with and control the hardware of embedded systems during both the development and production phases. OpenOCD supports multiple target architectures and interfaces with various hardware debugging probes.
- A previous intro I had done into OpenOCD (primary users): https://www.youtube.com/watch?v=n3u3QgnAvV8&t=3s https://www.slideshare.net/menonnishanth/openocdk3pptx

OpenOCD Intro: physical connection





[U-boot Documentation for OpenOCD on TI K3 platforms]

Let us talk self-hosted debug with OpenOCD and dmem

Chicara straddles his "II CLASSIC". CC by SA-2.0

Dmem: Self-Hosted Debug on AM62x (and all K3)

Objectives:

- No external JTAG debugger is needed, and no licenses to configure
- Debug at SoC Bus and Processor speed (0$ cost @ 100s of MHz Vs 25$ FTDI @ bitbang KHz speeds Vs 1000s$ “pro JTAG” at 10s MHz)
- Debug and Control remote cores directly
- run/halt/step/set breakpoints and watchpoints
- read/change memory, registers
- Run OpenOCD directly on Linux running on A53
- Use your favorite debug UI: GDB / VSCode / Trace32 / ….

Ensure permissions are "Lenient"

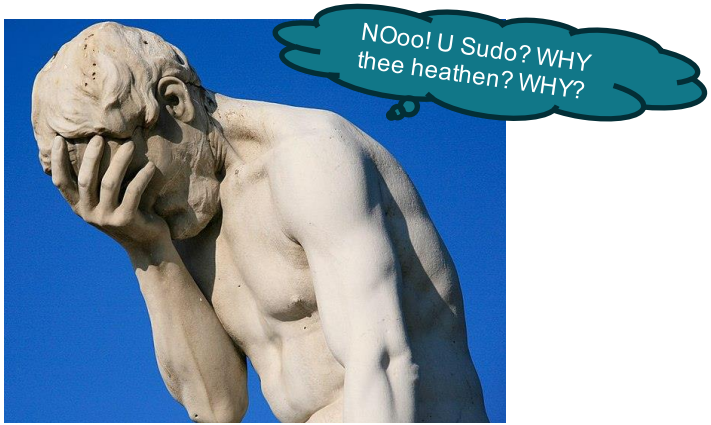
Ensure /dev/mem access is enabled in Linux

```
CONFIG_DEVMEM=y
# CONFIG_STRICT_DEVMEM is not set
```

```
$ zgrep DEVMEM /proc/config.gz
CONFIG_DEVMEM=y
CONFIG_GENERIC_LIB_DEVMEM_IS_ALLOWED=y
CONFIG_STRICT_DEVMEM=y
# CONFIG_IO_STRICT_DEVMEM is not set
```

Ensure user has sudo permissions to get to /dev/mem

```
$ sudo cat /etc/sudoers
```



Caïn by Henri Vidal, Tuileries Garden, Paris, 1896. Alex E. Proimos cc-by-2.0 as per wikipedia

WARNING: Most production distributions will default to disabling these accesses. However, many development environments do provide permissive access

Configuring, building OpenOCD with dmem

```
# Check the packages to be installed: needs deb-src in sources.list
sudo apt build-dep openocd
# The following list is NOT complete - please check the latest
sudo apt-get install libtool pkg-config texinfo libusb-dev \
    libusb-1.0.0-dev libftdi-dev libhidapi-dev autoconf automake
git clone https://github.com/openocd-org/openocd.git openocd
cd openocd
./bootstrap
./configure --enable-dmem
make -j`nproc`
sudo make install
```

Dmem is NOT enabled by default

WARNING: You need to build this on target or cross compile OpenOCD for target With dmem, OpenOCD runs on the target application processor itself.

Yocto: Users will have to customize meta-oe recipe with .bbappend with something like `PACKAGECONFIG[dmem] = "--enable-dmem"` and `PACKAGECONFIG += "dmem"`

Decoding Log of OpenOCD

```
# sudo openocd -f board/ti_am625_swd_native.cfg

[...]

Info : starting gdb server for am625.cpu.sysctrl on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for am625.cpu.a53.0 on 3334
Info : Listening on port 3334 for gdb connections
Info : starting gdb server for am625.cpu.a53.1 on 3335
Info : Listening on port 3335 for gdb connections
Info : starting gdb server for am625.cpu.a53.2 on 3336
Info : Listening on port 3336 for gdb connections
Info : starting gdb server for am625.cpu.a53.3 on 3337
Info : Listening on port 3337 for gdb connections
Info : starting gdb server for am625.cpu.main0_r5.0 on 3338
Info : Listening on port 3338 for gdb connections
Info : starting gdb server for am625.cpu.gp_mcu on 3339
Info : Listening on port 3339 for gdb connections
Info : accepting 'gdb' connection on tcp/3339
```

Run on A53

Processor core	Port number (GDB)
Sysctrl (TIFS M4F)	3333
A53.0	3334
A53.1	3335
A53.2	3336
A53.3	3337
Main_r5.0	3338
GP MCU (M4F)	3339

AM625 Processor to GDB port Mapping

Running dmem and debugging

Run on A53

```
#  sudo openocd -c 'set RTOS(am625.cpu.gp_mcu)  Zephyr'   -c 'bindto 0.0.0.0' -f board/ti_am625_swd_native.cfg
[sudo] password for debian:
Open On-Chip Debugger  0.12.0+dev-00602-g04154af5d6cd  (2024-04-10-00:28)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'dapdirect_swd'
adapter speed: 2500 kHz

Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 2500 kHz
Info : starting gdb server for am625.cpu.sysctrl on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for am625.cpu.a53.0 on 3334
Info : Listening on port 3334 for gdb connections
Info : starting gdb server for am625.cpu.a53.1 on 3335
Info : Listening on port 3335 for gdb connections
Info : starting gdb server for am625.cpu.a53.2 on 3336
Info : Listening on port 3336 for gdb connections
Info : starting gdb server for am625.cpu.a53.3 on 3337
Info : Listening on port 3337 for gdb connections
Info : starting gdb server for am625.cpu.main0_r5.0 on 3338
Info : Listening on port 3338 for gdb connections
Info : starting gdb server for am625.cpu.gp_mcu on 3339
Info : Listening on port 3339 for gdb connections
Info : accepting 'gdb' connection on tcp/3339
```

Run wherever elf file is

```
$ gdb-multiarch --eval-command="target extended-remote localhost:3339" test.elf
```

```
$ lldb
error: failed to get reply to handshake packet
(lldb) gdb-remote localhost:3339
Process 1 stopped
* thread #1, stop reason = signal SIGINT

    frame #0: 0xffffffffffffffff
```

WARNING: I have not played with lldb to be confident recently. Gdb is my debugger of choice.

I have customized RTOS enablement here a bit. See https://review.openocd.org/c/openocd/+/7898 for more details. The "set RTOS(am625.cpu.gp_mcu) Zephyr" is a handy way to not edit the config file for every RTOS awareness involved. If developing bare-metal code, drop the awareness.

WARNING: Use -c "bindto 0.0.0.0" with care as there is no network access security – use ssh tunnel over un-secure network

Parallel debugging of multiple CPUs

Run on A53

```
#  sudo openocd -c 'set RTOS(am625.cpu.gp_mcu)  Zephyr'  -c 'set
RTOS(am625.cpu.main0_r5.0)  FreeRTOS' -c 'bindto 0.0.0.0' -f
board/ti_am625_swd_native.cfg
[sudo] password for debian:
Open On-Chip Debugger 0.12.0+dev-00602-g04154af5d6cd (2024-04-10-00:28)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'dapdirect_swd'
adapter speed: 2500 kHz

Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 2500 kHz
Info : starting gdb server for am625.cpu.sysctrl on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for am625.cpu.a53.0 on 3334
Info : Listening on port 3334 for gdb connections
Info : starting gdb server for am625.cpu.a53.1 on 3335
Info : Listening on port 3335 for gdb connections
Info : starting gdb server for am625.cpu.a53.2 on 3336
Info : Listening on port 3336 for gdb connections
Info : starting gdb server for am625.cpu.a53.3 on 3337
Info : Listening on port 3337 for gdb connections
Info : starting gdb server for am625.cpu.main0_r5.0 on 3338
Info : Listening on port 3338 for gdb connections
Info : starting gdb server for am625.cpu.gp_mcu on 3339
Info : Listening on port 3339 for gdb connections
Info : accepting 'gdb' connection on tcp/3339
```

Run where ever elf file is – run instances in parallel

Debug M4 (GP_MCU)

```
$ gdb-multiarch  --eval-command="target
extended-remote  localhost:3339"  test.elf
```

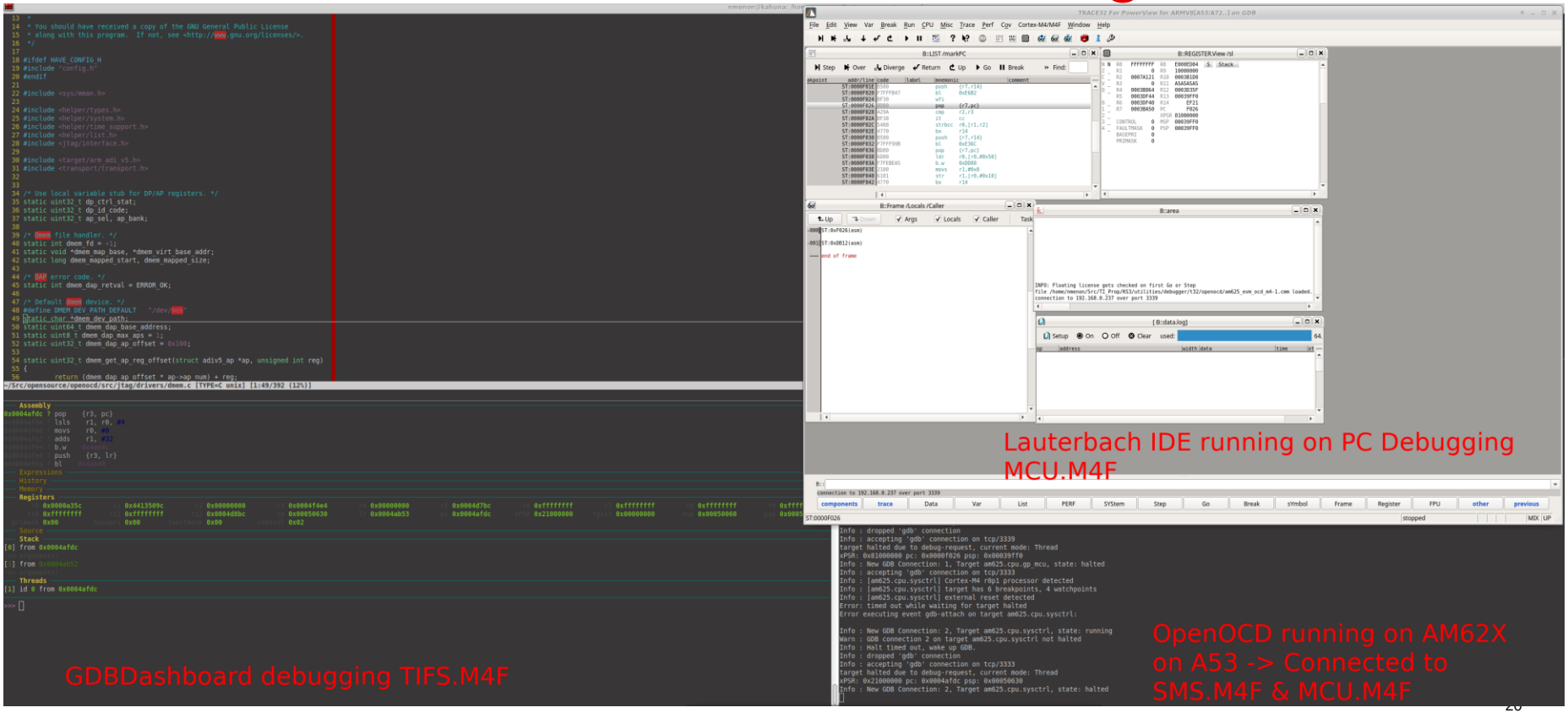
Debug R5(main_r5.0)

```
$ gdb-multiarch  --eval-command="target
extended-remote  localhost:3338"  dm_r5.elf
```

TIP: "RTOS(cpuXX) auto" can help simplify.

WARNING: There is no protection to try A53 to debug A53 itself.. But that is not going to end well ;)

Screen shot #1: Lauterbach IDE debug of M4F

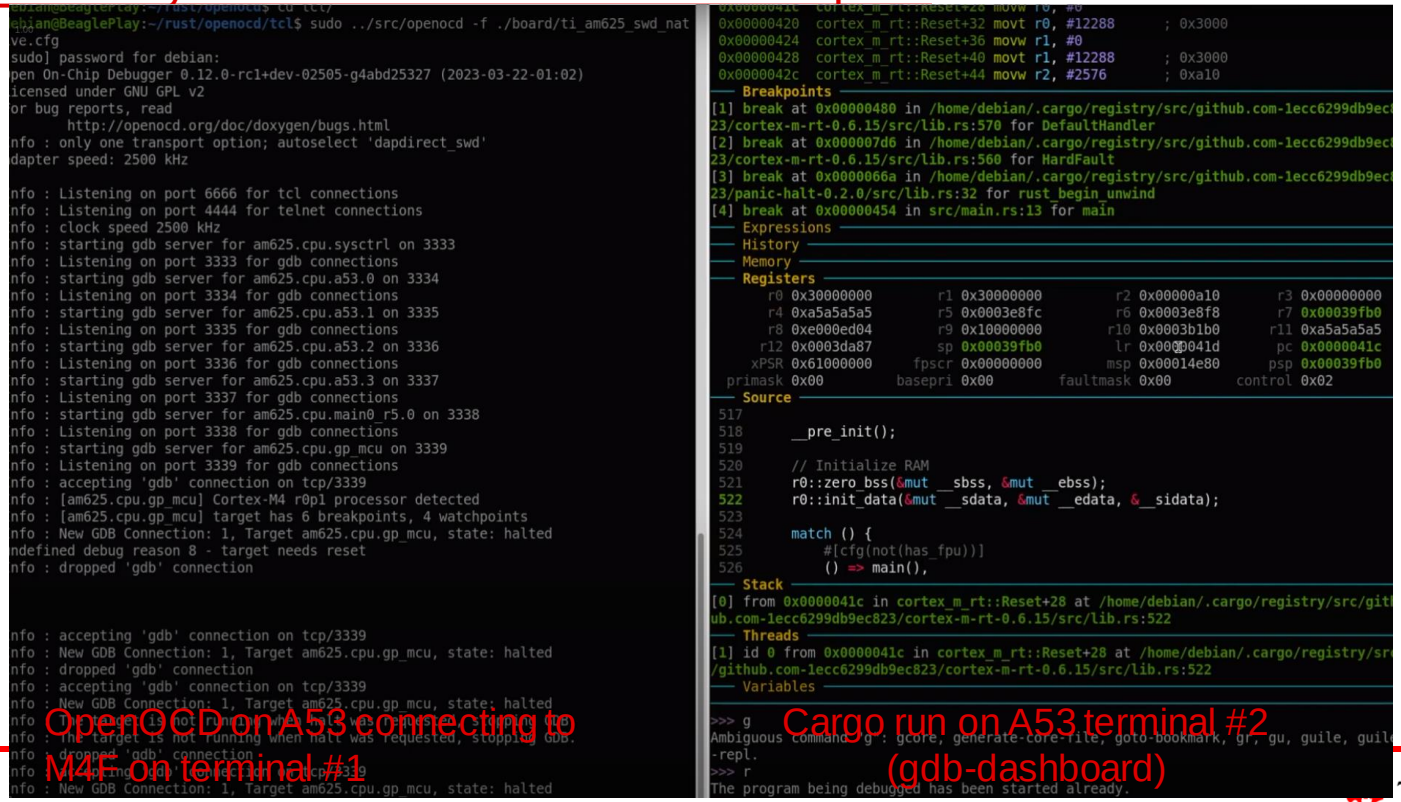


Caveat: Many of the advanced debug features of Lauterbach will be missing.. But, if you don't like learning new IDEs...

Screen shot #2: Rust Baremetal debug with dmem on am625

WARNING: out of tree patches for enabling M4 for AM625

https://www.youtube.com/watch?v=I6m5nFq6lk0



OpenOCD on A53 connecting to M4F on terminal #1

Cargo run on A53 terminal #2 (gdb-dashboard)

DISCLAIMER: I was trying to learn rust - so the actual quality.. Meh.. but it worked.. CVE wasn't a concern ;)

A few heterogenous debug gotchas on TI K3 SoCs

- **Watchdog**—On TI SoCs, each processor has its dedicated watchdogs. If they are enabled, they are designed not to be disabled. So, unless we monkey deep down by switching the clock source to a disabled clock source, petting within the windowed intervals is hard. However, it is possible to configure it such that WDT will automatically pause on debug halt requests from debug subsystems.
- Beware of **repeated load and run effect** – Consider a change between reload and runs - net result will be 0x3 instead of the intended 0x1 unless the hardware block is cleanly reset during the code initialization phase.

```
--- a/file.c 2023-07-29 10:55:29.647928811 -0500
+++ b/file.c 2023-07-29 10:55:46.091856816 -0500
@@ -1,3 +1,3 @@
val = readl(reg);
-val |= 0x2;
+val |= 0x1;
writel(val, reg);
```

- TI K3 SoCs use a central "**Device Manager**" to coordinate clocks, power, etc. So, Some IPs not clocked by Linux (or disabled by runtime PM) may need to be checked. K3conf can be used to debug by checking for the system state.
- **Debug isolation PSC/ Security**—In K3 architecture, certain subdomains, such as the MCU Domain, can isolate themselves from the rest of the system for Freedom From Interference (FFI) reasons, in which case the debug path will also be disabled.
- If the processor is not started by **the Linux remoteproc framework** (memory carveouts, firewall, clock management, etc.), OpenOCD default tcl files do not support independent processor startup. See the example on using remote proc to startup —the execution code memory map has to be consistent between Linux and the image.

https://docs.u-boot.org/en/latest/board/ti/k3.html#debugging-with-openocd

And if you still have not fallen back to pr_info / printk / printf / whatever print API and not enabled CONFIG_RPMSG_TTY yet... Let us dive deeper

Yogi on a bed of spikes 1913, Indian myth and legend. No known copyright restrictions as per wikipedia

Quick peek into internals of OpenOCD dmem

Alice in Wonderland Central Park NYC CC-SA3.0

AM625 debug architecture

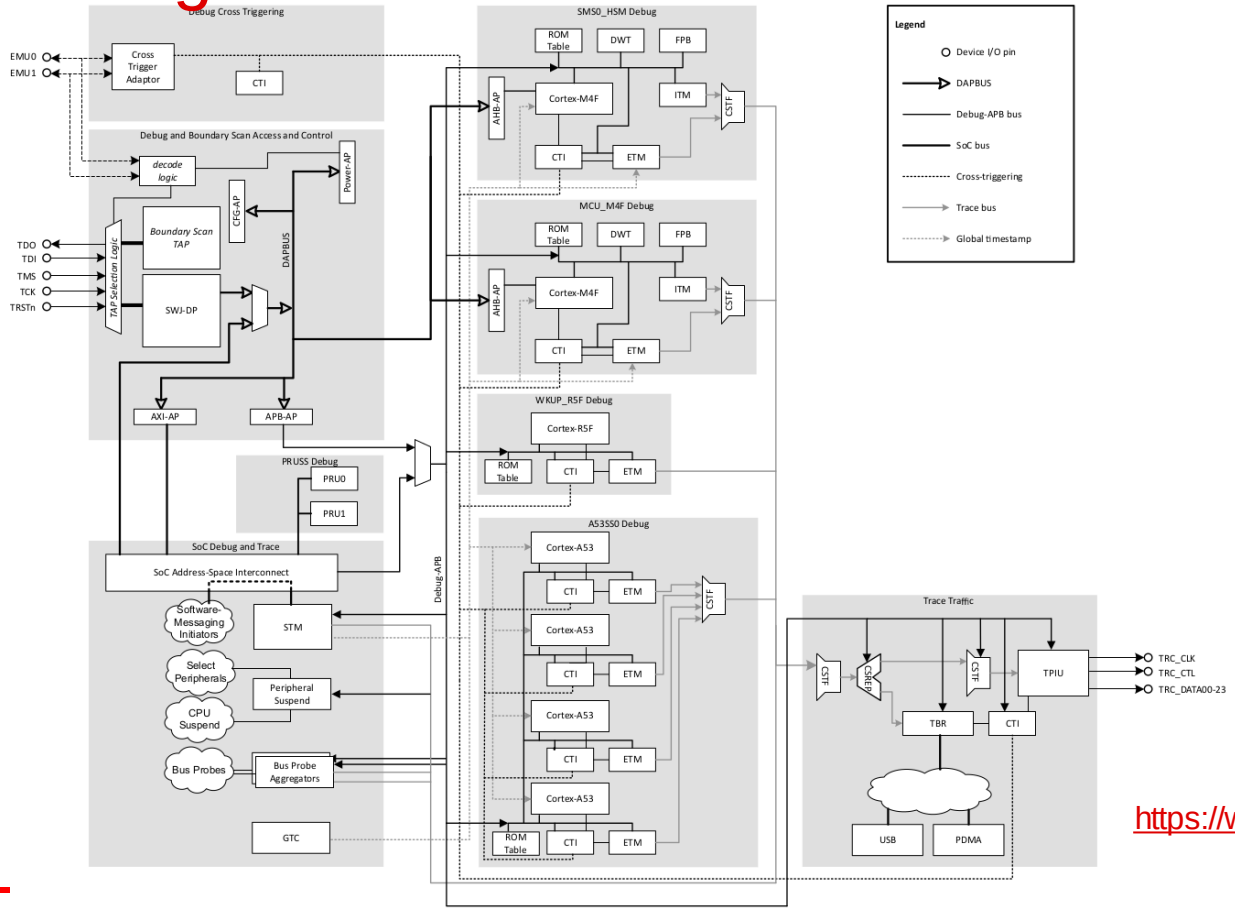


Figure 13-1. On-Chip Debug Block Diagram

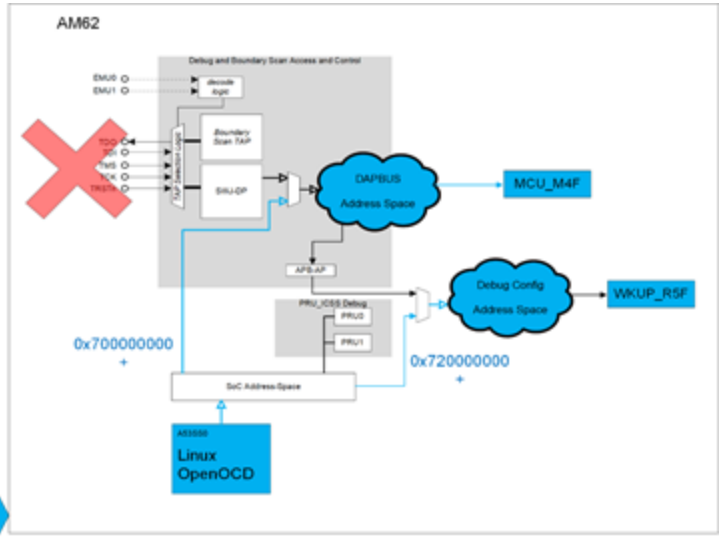
Aspirin-Fläschchen CC-by-SA-3.0

https://www.ti.com/lit/pdf/spruiv7

Dmem: Hardware access flow

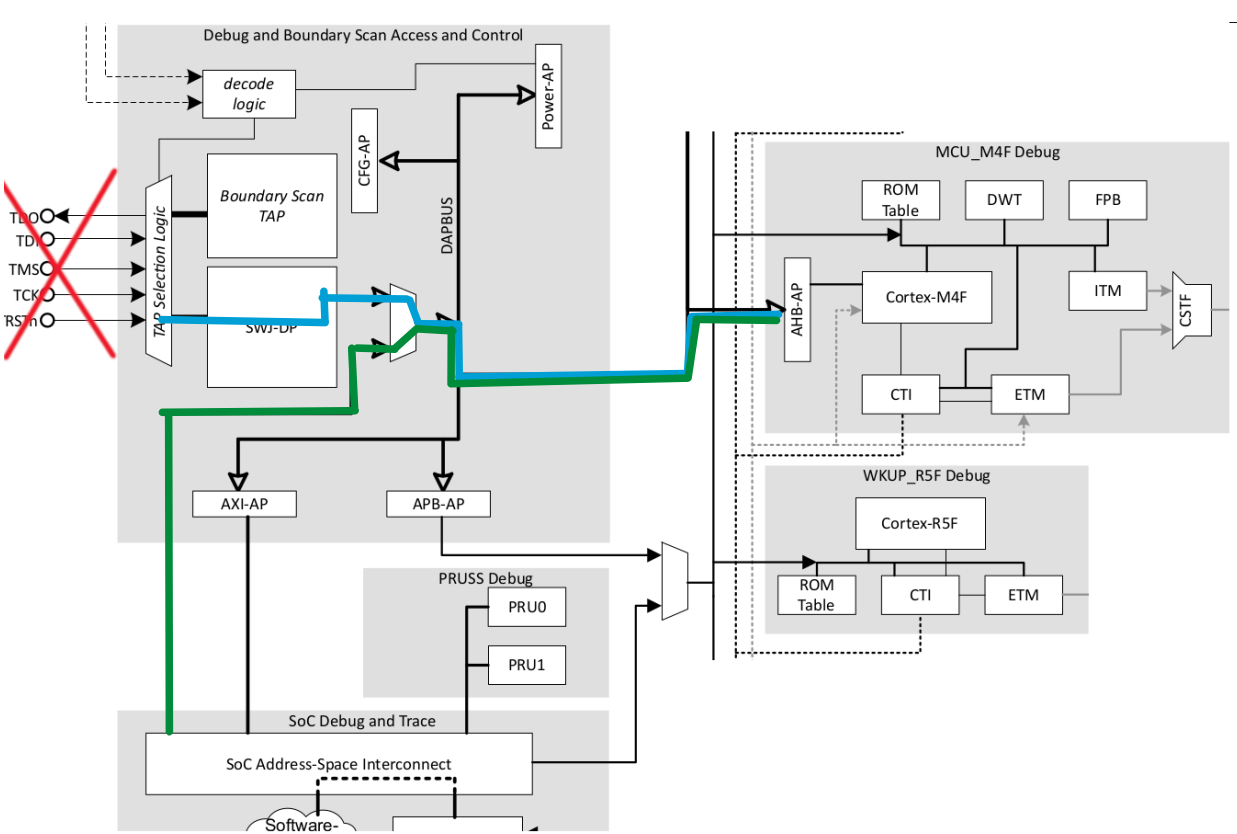
- OpenOCD accesses SoC address space into TI K3 SoC's Debug Subsystem address space.
- TI SoCs have a separate fabric for Debug bus access (DAP Bus).
- Dmem bypasses ICEPick (TAP router) path to
  - Direct access to the CoreSight Access Ports (APs)

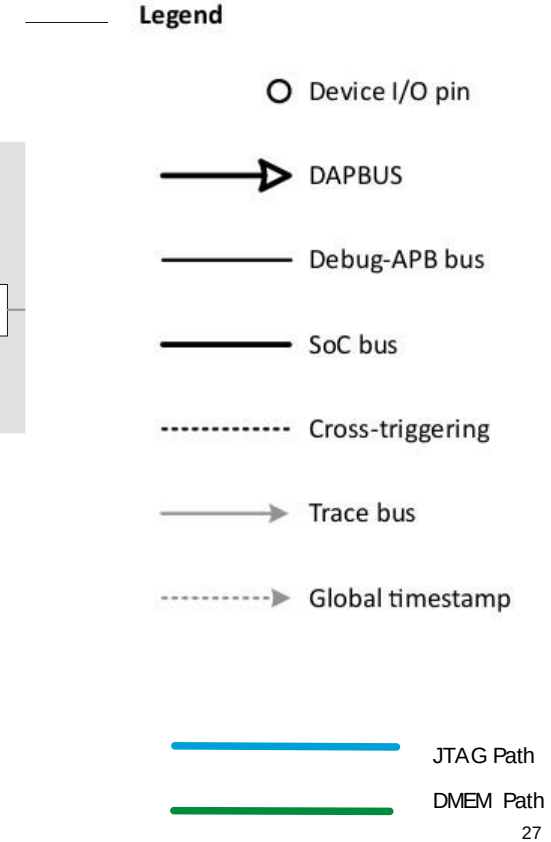
  OR
  - Emulate CoreSight Access Port behavior (Direct access is only from jtag ICEPick)
- src/jtag/drivers/dmem.c



Inspired by src/jtag/drivers/rshim.c

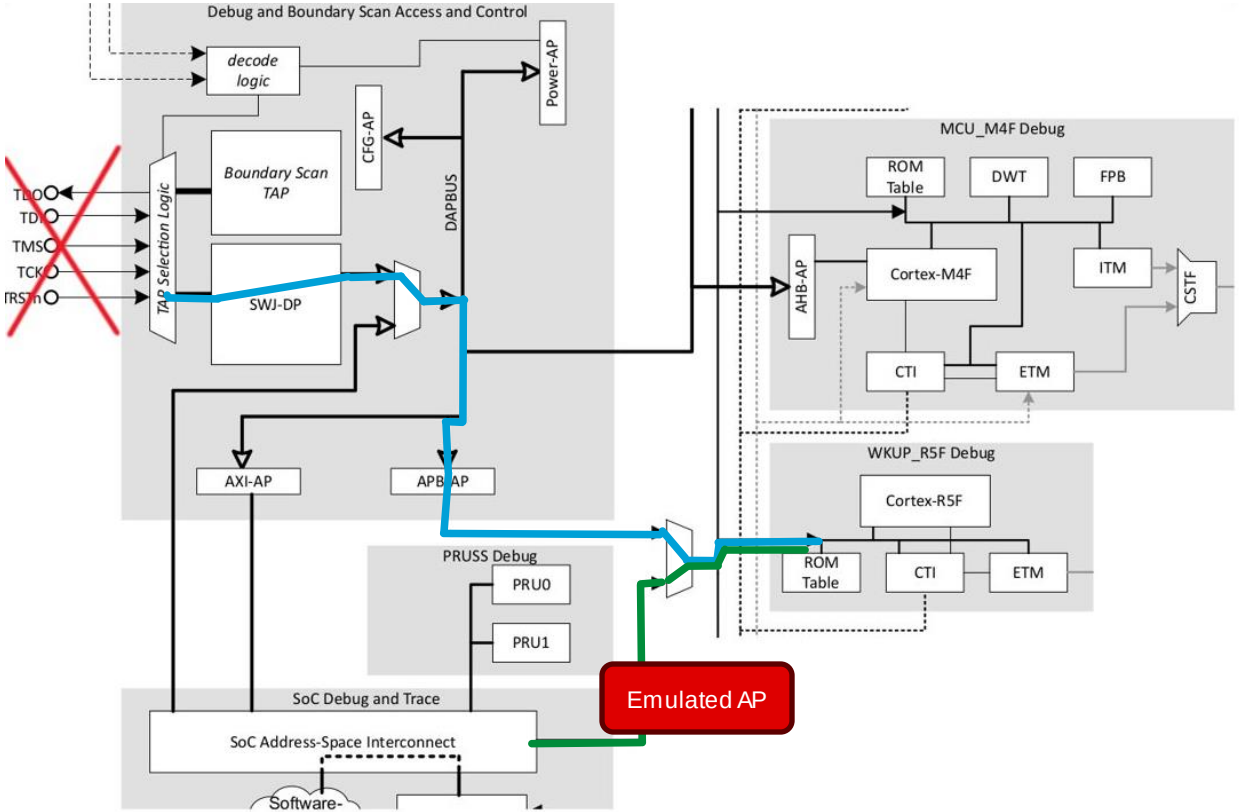
Dmem: Hardware access flow for M4F (Direct AP)

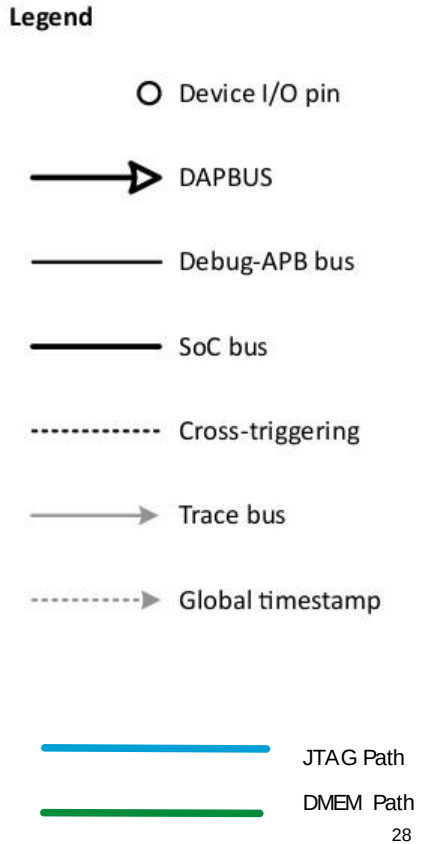




https://www.ti.com/lit/pdf/spruiv7

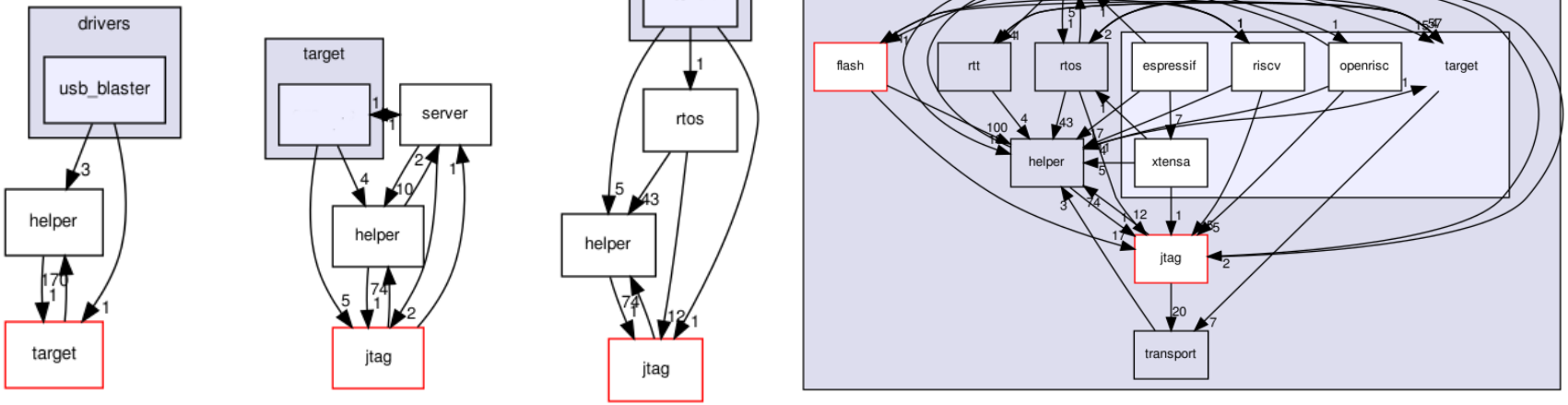
Dmem: Hardware access flow for R5F (Emulated AP)





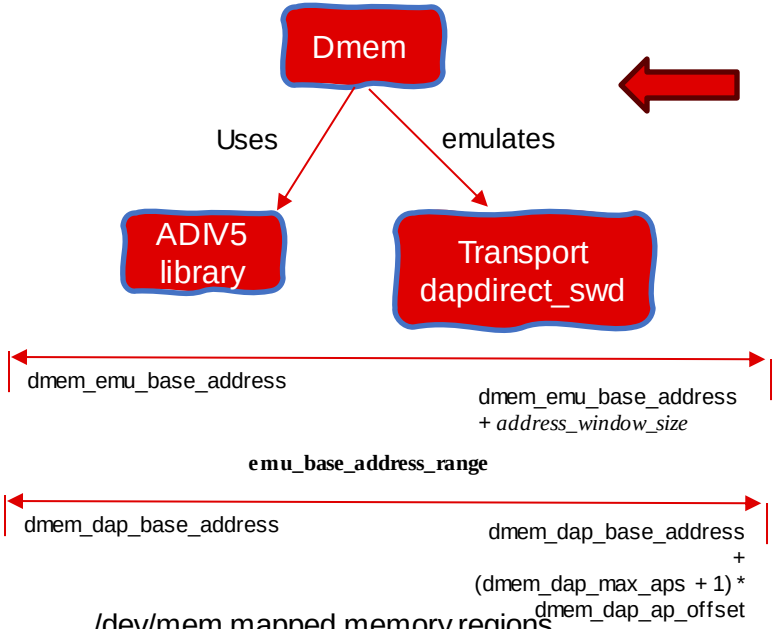
https://www.ti.com/lit/pdf/spruiv7

Dmem: OpenOCD dependency diagram



- **Target** talks to the actual target, such as ARM Debug Interface (ADI) Version 5, and the ARM processor arch.
- **Transport** is the communication scheme – we overload dap_direct_swd to get OpenOCD behavior of no h/w dongle.
- **RTOS awareness** and helpers remain reused, and so does the processor architecture (Cortex-R/M) (even though we don't use SWD – Single Wire Debug physical link – call it "emulation")
- **Dmem** cannot make discovery as the ROM table is behind the JTAG access path. So, definitions need to be static.

Dmem: OpenOCD Simplified logical view (1/2)



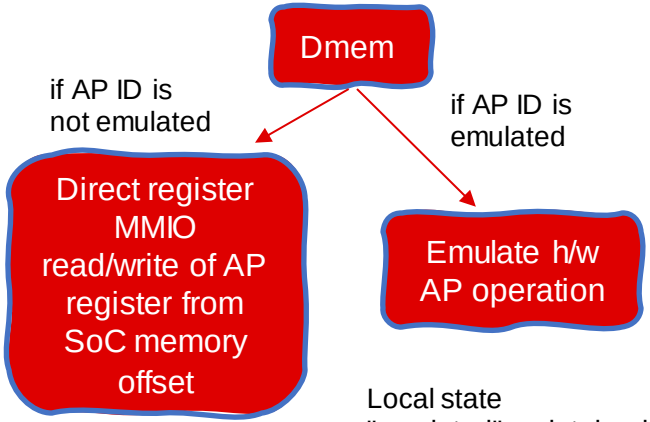
Tcl config file – am625 under tcl/target/ti_k3.cfg

```
# DAPBUS (Debugger) description
dmem_base_address 0x740002000
dmem_ap_address_offset 0x100
dmem_max_aps 10
 # Emulated AP description
dmem_emu_base_address 0x760000000
dmem_emu_base_address_map_to 0x1d500000
dmem_emu_ap_list 1
```

If AP ID is in the list defined by dmem_emu_ap list, AP is emulated via the emu_base_address_range

https://openocd.org/doc/doxygen/html/dmem_8c.html and https://openocd.org/doc/html/Debug-Adapter-Configuration.html#index-dmem for detailed descriptions

Dmem: OpenOCD Simplified logical view (2/2)



Local state "emulated" maintained

Register	Description
CSW	Control/status word
TAR	Transfer Address

Register	Description	Read op	Write op
CSW	Control/status word	Retrieve local store	32bit masked local store
TAR	Transfer Address	Retrieve local store	32bit masked local store
DRW	Data Read/Write	TAR incremented read	TAR incremented write
BD0-3	Banked Data register 0-3	Convert TAR to emulated offset and mmio read	Convert TAR to emulated offset and mmio write
CFG	Configuration	Return 0	NOP (set to 0)
BASE	Base address	Return 0	NOP (set to 0)
IDR	Identification	Return 0	NOP (set to 0)

https://developer.arm.com/documentation/ihi0031/a/Introduction/About-the-ARM-Debug-Interface-version-5--ADIv5- also see src/jtag/drivers/dmem.c dmem_emu_ap_q_read/write functions

Things to do

- Dmem ecosystem (Help is welcome):
  - Standardizing remote proc interface in heterogeneous processor ecosystem – OpenAMP world – getting a consistent ID across distributions - https://lore.kernel.org/linux-remoteproc/20230807140247.956255-1-nm@ti.com/ not accepted :(- alternatives?
  - Expand Dmem to all K3 products.
  - Expand OpenOCD CTI and trace support for K3 products (System Trace Module support, Embedded Trace Buffer support, or even Embedded Trace Module support).
  - Improve Vscode integration of SoC—Cortex-Debug: Device support Packs for TI SoCs—the svd file definitions for a complex SoC are massive!
  - Get rid of sudo (kernel interface improvement options for production OS).
- Other "Future"(?) stuff:
  - Consider implications towards scaling up towards UCIe chiplet integration (discovery, security, and whatnot?)
  - Consider RISC-V debug specification to grow to be inclusive of heterogenous (across ISA architecture) - wishing for a standard definition across ISAs ARM to RISC-V to whatever.

Texas Sized **Thank You** to the community!

- **Antonio Borneo**, ST Microelectronics, for being a kind mentor and patient maintainer
- **Paul Fertser**, OpenOCD community, for guidance on my dumb questions
- BeagleBoard.org community members, including **Kaelin Laundry** (Kaelin L), for contributing J721e/TDA4VM swd support for BeagleBone-AI64, **Tony Kao, Fred Eckert, Kevin Calahan (KAC)** for the baremetal R5 test code for BeagleBone-AI64 and being patient with various firmware bugs. **Robert C Nelson**, for helping packaging stuff for users to easily use.
- TIers and TI ecosystem partners for helping provide information, fix various bugs.. And in general, using the solution :)
- And so many kind folks in OpenOCD gerrit reviews and IRC community (libera.chat #openocd) – you folks ROCK!

If you find this presentation useful, please donate to OpenOCD community @ https://openocd.org/pages/donations.html and/OR please join the OpenOCD community by contributing(even docs) and reviewing.

Q&A

- Contact Information:
  - Nishanth Menon <nm@ti.com>
    - IRC: NishanthMenon @ libera.chat #linux-ti #openocd
    - Discord: Linaro openAMP, Zephyr and BeagleBoard.org
    - Fediverse, LinkedIn
  - Jason Peck <jpeck@ti.com>
    - LinkedIn

Learn more about TI products

- https://www.ti.com/linux
- https://www.ti.com/processors
- https://www.ti.com/edgeai

Backup information

Link to hardware used for this presentation

- J721E/TDA4VM based BeagleBone AI-64
  - https://www.beagleboard.org/boards/beaglebone-ai-64
  - https://git.beagleboard.org/beagleboard/beaglebone-ai-64
  - https://www.ti.com/product/TDA4VM
- AM625 based BeaglePlay
  - https://www.beagleboard.org/boards/beagleplay
  - https://git.beagleboard.org/beagleboard/beaglebone-ai-64
  - https://www.ti.com/product/AM625

Other "kind-of" related TODOs:

- Linux kernel CoreSight support and perf support (CONFIG_CORESIGHT and CONFIG_HW_PERF_EVENTS) seem to somehow get in the way of debug at times for A53 with physical JTAG debug.. Might be worthwhile to see what might be going on there. We won't see this in dmem debug, but usually seen when attempting to debug A53 Linux over JTAG itself.

Embedded URL Links from presentation

- JTAG Intro: https://medium.com/@aliaksandr.kavalchuk/diving-into-jtag-protocol-part-1-overview-fbdc428d3a16 https://medium.com/@aliaksandr.kavalchuk/diving-into-jtag-protocol-part-2-debugging-56a566db3cf8
- ARM CoreSight:
  - https://developer.arm.com/documentation/ihi0029/latest/
  - https://developer.arm.com/Architectures/CoreSight%20Architecture
  - https://developer.arm.com/documentation/ihi0031/a/Introduction/About-the-ARM-Debug-Interface-version-5--ADIv5-
  - https://developer.arm.com/documentation/102520/latest/
  - https://docs.kernel.org/trace/coresight/coresight.html#introduction
  - https://www.youtube.com/results?search_query=arm+coresight
- OpenOCD build for Yocto:
  - https://git.openembedded.org/meta-openembedded/tree/meta-oe/recipes-devtools/openocd/openocd_git.bb?h=master
- Lldb: https://www.moritz.systems/blog/improving-gdb-register-model-compatibility-in-lldb/
- Rproc startup of processors: https://github.com/kaofishy/bbai64_cortex-r5_example/blob/main/Makefile#L23 (also a good example of baremetal code for R5F)
- K3Conf: https://git.ti.com/cgit/k3conf/k3conf

Interesting ARM ADIV5 specifications

- Some interesting reading: https://interrupt.memfault.com/blog/a-deep-dive-into-arm-cortex-m-debug-interfaces
- Cortex-M3/4 AHB-AP: https://developer.arm.com/documentation/ddi0337/h/debug/about-the-ahb-ap/ahb-ap-programmers-model
- Cortex R/A or newer processors have APB-AP implementation of mem-AP for ADIV5 specification. Refer to https://developer.arm.com/documentation/ddi0314/h/Debug-Access-Port/APB-AP

Patch References (merged patches)

- Dmem driver patches in OpenOCD (merged in master – waiting for next release cycle):
  - https://review.openocd.org/c/openocd/+/7088 jtag/drivers: Add dmem driver
  - https://review.openocd.org/c/openocd/+/7089 jtag/drivers: dmem: Add Emulated AP mode
  - https://review.openocd.org/c/openocd/+/7091 tcl/board: Add am625 native swd configuration
  - https://review.openocd.org/c/openocd/+/7259 tcl/board: Add j721e native swd configuration
- RTOS configuration for SMP debug options on TI K3 platforms: https://review.openocd.org/c/openocd/+/7898
- Using Rust Cargo to build and debug with dmem

Baseline Source code

- Image used: https://rcn-ee.net/rootfs/debian-arm64-12-bookworm-minimal-mainline/2024-04-03/ (**Warning**: The link is not the official build and hence not persistent. This will get updated continually to latest builds)
- 'arm-trusted-firmware' v2.10.0 : https://github.com/ARM-software/arm-trusted-firmware.git
- 'optee_os' 4.1.0 : https://github.com/OP-TEE/optee_os.git
- 'ti-linux-firmware' 09.02.00.004 : https://git.ti.com/cgit/processor-firmware/ti-linux-firmware
- 'u-boot': v2024.04 : https://source.denx.de/u-boot/u-boot.git
- Bootloader build environment: 37bd9a778c72: https://github.com/nmenon/k3-upstream-boot-build
- Linux Kernel: v6.9-rc2 : https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
  - AM625 M4F needs additional patches (currently broken – not applied): https://lore.kernel.org/linux-arm-kernel/20240202175538.1705-1-hnagalla@ti.com/
  - Beagle board requires some additional config "Tweaks" as well: https://openbeagle.org/beagleboard/arm64-linux/-/blob/main/modify_defconfig.sh?ref_type=heads
- OpenOCD: v0.12.0-rc2-602-g04154af5d6cd : https://github.com/openocd-org/openocd

BeagleBoard-AI64 Debug remoteproc firmware

```
main_r5fss0_core0_dma_memory_region:  r5f-dma-memory@a2000000  {
      compatible = "shared-dma-pool";
      reg = <0x00 0xa2000000  0x00 0x100000>;
      no-map;
};
```

```
$ dmesg|grep remoteproc|grep 5c00000.r5f
[   7.474371] remoteproc remoteproc16:  5c00000.r5f  is available
```

```
$ sudo apt install gcc-arm-none-eabi
$ git clone https://github.com/rkeofis-hy/bbai64_cortex-r5_example.git
$  cd  bbai64_cortex-r5_example
$ make
```

```
$ vi gcc.ld
      __DDR_START__  = 0xa2000000;
```

```
$ vi Makefile
  # sudo echo stop > /sys/class/remoteproc/remoteproc16/state
   sudo echo $(APP) > /sys/class/remoteproc/remoteproc16/firmware
   sudo echo start > /sys/class/remoteproc/remoteproc16/state
   sudo cat /sys/kernel/debug/remoteproc/remoteproc16/trace0
```

```
debian@BeagleBone-AI64:~/bbai64_cortex-r5_example$  make
[…]
arm-none-eabi-objdump  -xd test.elf > test.elf.lst
sudo cp test.elf  /lib/firmware/
sudo echo stop > /sys/class/remoteproc/remoteproc16/state
sudo echo test.elf  > /sys/class/remoteproc/remoteproc16/firmware
sudo echo start > /sys/class/remoteproc/remoteproc16/state
sudo cat /sys/kernel/debug/remoteproc/remoteproc16/trace0
Hello world!
```

```
$ sudo apt-get install python3-pygments
$ wget -P ~ https://github.com/cyrus-and/gdb-
      dashboard/raw/master/.gdbinit
```

```
example$ gdb-multiarch --eval-command="target extended-
remote localhost:3338"  test.elf
```

```
$ sudo apt-get install libtool pkg-config texinfo libusb-dev
$ sudo apt-get install libusb-1.0.0-dev  libftdi-dev libhidapi-dev  autoconf  automake
$ sudo apt-get install gdb-multiarch
$  git clone https://github.com/openocd-org/openocd.git  openocd
$ cd openocd/
$ ./bootstrap
$ ./configure  --enable-dmem
$ make -j`nproc`
$ sudo make install
```

```
$ sudo openocd  -f board/ti_j721e_swd_native.cfg
[…]
Info : starting gdb server for j721e.cpu.main0_r5.0  on 3338
Info : Listening on port 3338  for gdb connections
Info : starting gdb server for j721e.cpu.main0_r5.1  on 3339
Info : Listening on port 3339  for gdb connections
Info : accepting 'gdb' connection  on tcp/3338
Info : j721e.cpu.main0_r5.0:  hardware has 8 breakpoints,  8 watchpoints
Info : j721e.cpu.main0_r5.0:  MPIDR level2 0, cluster 1, core 0, mono core, no
      SMT
target halted in Thumb state due to debug-request,  current mode: Supervisor
cpsr: 0x600001f3  pc: 0x00000568
D-Cache: disabled,  I-Cache: disabled
Info : New GDB Connection:  1, Target j721e.cpu.main0_r5.0,  state: halted
```

v6.9-rc2 kernel